



SECURITY ADVISORY

Wasmtime sandbox escapes

CVE-2026-115889 · CVE-2026-115893

Assessment & Action Plan

Two critical Cranelift bugs let a crafted WebAssembly module escape the sandbox. Which fixed version is yours, who is exposed, and what to do this week.

PREPARED BY	Hung Le, NamiQ	DATE	25 October 2026
FACTS AS OF	25 October 2026 (advisories of 19 October)	VERSION	1.0
SEVERITY	Critical: CVSS 3.1 9.9 (both)	STATUS	Fixed; see section 3

NamiQ © 2026 NamiQ. All rights reserved. This document is the property of NamiQ. It may be shared unaltered, with attribution.

Companion article: namiq.io/insights/wasmtime-cranelft-cve-2026-115889-115893

1. The short version

- **CVE-2026-115889** (GHSA-897f-g5gm-fmxm): Cranelift packs value indices into bit fields that can overflow silently in release builds. A very large or crafted function body can cause type confusion and miscompilation. **All architectures**, default settings.
- **CVE-2026-115893** (GHSA-mghf-wjfp-p5xm): on **aarch64**, a `br_table` with more than about 256,000 targets compiles to a conditional branch with the wrong target, so execution can jump into other code.
- Both end in a sandbox escape. Neither advisory lists a workaround; the remedy is to upgrade.
- Neither advisory reports exploitation. Check the CISA Known Exploited Vulnerabilities catalog before relying on that.

2. Severity decoded

Both carry `CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H`, score 9.9.

Metric	Value	What it means here
Attack vector	Network	The score assumes modules arrive over a network, as in a platform that accepts uploaded code.
Complexity	Low	The advisory rates the attack as not needing special conditions beyond the crafted module.
Privileges	Low	The attacker must be able to get a module compiled by your engine. That is the real gate.
User interaction	None	No one has to click anything.
Scope	Changed	The damage leaves the sandbox and reaches the host. This is what lifts the score to 9.9.
C / I / A	High / High / High	Full compromise of the host process and what it can reach.

Read the privilege line before the score. If every module you run is one you built and control, the attacker has no way in, and your practical exposure is small however high the number is.

3. Which version fixes which CVE

Release line	CVE-2026-115889 (all architectures)	CVE-2026-115893 (aarch64 only)
Below 36.0.19	Affected. Fixed in 36.0.19.	Affected. Fixed in 36.0.19.
36.0.19	Fixed	Fixed
37.0.0 to 42.x	Affected. No fix listed for these lines.	Affected. Move to 48.0.7 or later.
43.x to 47.x	Not affected (fixed in 43.0.0).	Affected. Move to 48.0.7 or later.
48.x	Not affected	Fixed in 48.0.7
49.x	Not affected	Fixed in 49.0.4

A single number such as "update to 36.0.19 or 43.0.0" is correct for 115889 only. On aarch64, 43.0.0 does not fix 115893.

Claims in circulation that are wrong

- **"Switch to the LLVM backend."** We found no such option in the advisories, which list no workaround. Do not plan around it.
- **"Both are aarch64 bugs."** Only 115893 is.
- **"One fixed-version pair covers both."** See the table.

4. Exposure by deployment

Your situation	Priority	Timeframe
Untrusted or customer WebAssembly, any architecture	High (115889)	Upgrade within days
Untrusted WebAssembly on aarch64 (Apple Silicon, Graviton, other Arm servers)	High (both)	Upgrade within days
Only modules you build and control	Medium	Next regular release cycle
Do not run Wasmtime or anything embedding it	None	Record for audit

These timeframes are our judgement from the attacker prerequisites, not figures from the advisories.

5. Per-host procedure

Step 1: find where Wasmtime runs

```
# Binaries and libraries
wasmtime --version
find / \( -name 'wasmtime*' -o -name 'libwasmtime*' \) -not -path '*/proc/*' 2>/dev/null

# Embedded copies: Rust, Python, Go, container images
grep -rnE 'wasmtime' --include=Cargo.lock --include=Cargo.toml --include='requirements*.txt' --include=go.mod --include=Dockerfile .

# Architecture of each host
uname -m
```

Check CI runners, Arm build hosts and any platform you operate that runs customer or plugin WebAssembly. A statically linked binary will not appear in a package list, so search lockfiles too.

Step 2: upgrade to the version in section 3

- Rust: raise the `wasmtime` dependency, rebuild, redeploy.
- Other languages: update the wrapper package that pins Wasmtime, and confirm what version it bundles.
- Verify the version from the running service, not from the lockfile.
- Add a dependency check that fails builds below the fixed baseline.

Step 3: regression test

Run your WebAssembly test suite. Fixes in a compiler backend can change generated code, so run the workloads that matter to you, not only a smoke test.

6. If you cannot upgrade this week

Control	What it does	Cost
Stop compiling modules from sources you do not control	Removes the attacker's way in.	Feature loss until the upgrade lands.
Run the engine in its own low-privilege process, microVM or container, no secrets, no outbound network	An escape lands somewhere with nothing to take.	Engineering time; some latency.
Reject oversized modules at the loader	Both bugs need an unusually large function or branch table. A heuristic, not a fix.	May reject legitimate large modules.

7. Hunting checklist

- Modules with a function body or `br_table` far larger than anything your users normally submit.
- Engine processes that crash, or that start shells, download tools or open outbound connections.
- Submissions from new accounts that include a very large module shortly before an engine crash.

8. Definition of done

- Every host running Wasmtime is listed, with its architecture and release line.
- Every one runs the fixed version for its line, verified from the running service.
- A dependency check fails builds below that baseline.
- If you could not upgrade, the engine runs isolated and untrusted modules are blocked, and a date is set to remove the exception.

9. What we do not know yet

- Whether anyone is exploiting either bug. Neither advisory reports it.
- Whether public exploit code appears.
- Which downstream products bundle an affected Wasmtime, and when they ship fixes.

Sources

- GHSA-897f-g5gm-fmxxm, CVE-2026-115889:
github.com/bytedcodealliance/wasmtime/security/advisories/GHSA-897f-g5gm-fmxxm
- GHSA-mghf-wjfp-p5xm, CVE-2026-115893:
github.com/bytedcodealliance/wasmtime/security/advisories/GHSA-mghf-wjfp-p5xm
- CISA Known Exploited Vulnerabilities catalog: cisa.gov/known-exploited-vulnerabilities-catalog